# EAX' Cipher Mode (May 2011)

Authors Avygdor Moise, Edward Beroset, Tom Phinney, Martin Burns, *Members, ANSI C12 SC17 Committee*

***Abstract*— We propose a block-cipher mode of operation that optimizes protection in small embedded automation devices that have both extremely large and extremely small messages, in which the canonical form of message addressing information, needed in forming nonces, has almost unbounded length.**

## I. Introduction

The ANSI C12.22 Protocol Specification for Interfacing to Data Communication Networks [1] utilizes the EAX' (EAX-prime) Cipher Mode. The motivation for this work was the somewhat unique requirements of supervisory control and data acquisition (SCADA) messaging associated with Automated Meter Reading (AMR) that operate in the context of an Advanced Metering Infrastructure (AMI), the principle use of this standard. However, these unique requirements may be applicable to many small embedded devices communicating in SCADA environments.

The developers of the standard were aware of several challenges for the protection of very large and very small and repetitive messages that conform to the ANSI C12.22 standard, due largely to the almost unbounded length of the message addressing information specified by the associated protocol, which must be used in nonce formation when applying this proposed mode to that messaging. This paper introduces this Cipher Mode and its capabilities and advantages.

CCM is a NIST-standardized Authenticated Encryption with Associated Data (AEAD) cryptographic mode developed for IEEE 802.11, now also used by IEEE 802.15.4 [7]. This mode has been suggested for similar applications to ANSI C12.22. However, several obstacles to the use of this mode have been identified with respect to messaging specific to ANSI C12.22. These shortcomings have been addressed in part by the EAX mode description [3], and further by the present EAX' mode. Additionally, CCM is constrained by its definition to applicability only with the AES-128 block cipher, a constraint relaxed in EAX and EAX'.

GCM is a second NIST-standardized parallelizable Authenticated Encryption with Associated Data (AEAD) cryptographic mode developed for applications requiring high-speed cryptography [8]. This mode also has been suggested for similar applications to ANSI C12.22. However, significant obstacles to the use of this mode also have been identified with respect to messaging specific to C12.22. These shortcomings have been addressed in part by the EAX mode description [3], and, further by the present EAX' mode.

This paper introduces these issues, their resolutions, and the description of EAX'. The description of EAX' was extracted from ANSI C12.22 [1].

Note that EAX' is described herein with references to using AES-128 as the block cipher. However, the EAX' mode could be utilized with other underlying block ciphers. The term EAX' implies that it is a derivative of EAX, utilizing a common mathematical notion of the suffix "'".

## II. EAX' Properties Relative To CCM Properties

The NIST version of CCM has four functional restrictions (which EAX lacks) and one common implementation restriction that make it less appropriate for use in ANSI C12.22:

**Nonce length:** CCM has a fixed maximum nonce input size of 13 bytes. The ANSI C12.22 Message format requires use of ApTitles [9] [10] and other information that may exceed 50 bytes, all of which must be included in the nonce in a manner unpredictable to an attacker. EAX directly processes nonces of arbitrary and variable size.

**Nonce unpredictability:** CCM generates nonces that are predictable to an attacker, making it trivial for an attacker to determine that two distinct messages under the same key are using the same nonce value. This makes it imperative that the nonce construction never duplicate a nonce value, even after the compression step that would be required to reduce the large ANSI C12.22 nonce inputs to the required CCM 13-byte nonce. EAX computes a nonce value that is unpredictable to an attacker, in a range of $2^{128}$ nonce values, so no special protection need be taken against duplicate nonce values (provided that the inputs to the nonce computation do not duplicate).

**"Online" capability: "**CCM requires that the entire message be available before authentication and plaintext encryption can begin, requiring that an entire message be buffered before CCM can be applied. ANSI C12.22 can have message sizes of many thousand bytes, with maximum sizes exceeding 60,000 bytes. Together these features make it difficult to use add-on hardware or software for existing ANSI C12.22 implementations, to minimize time to initial deployment of the new standard. EAX has "online" capability, which makes it suitable for use in add-on hardware or software serial link encryptors.

**Authenticate-before-decrypt:** CCM requires that the entire received message be decrypted before it can be authenticated. Thus the extra processing required to decrypt Ciphertext must be applied whether the received message authenticates or not. EAX authenticates the Ciphertext rather than the plaintext, so this extra processing is required only for messages that have

been authenticated. This reduces the ability of an attacker to mount a DoS (denial of service) attack on the receiving device, causing it to consume significant resources before determining that the received message is invalid.

**Hardware non-support of canonicalized messages:** Some modem chips such as many of those for IEEE 802.15.4 provide hardware support for CCM-mode transmission and reception when the message to be CCM-authenticated consists exactly of the message Cleartext and plaintext, in that order, that is to be sent or that was received. However, ANSI C12.22 requires that the authenticated message use canonical forms of ApTitles, and exclude the <calling-AP-title-element> (the ApTitle of the message originator) when it was added by a proxy C12.22 Relay. The resultant software-based sequencing of invocations of the underlying hardware block cipher (e.g. AES-128) implementation is equivalent to that required for EAX.

## III. JUSTIFICATIONS FOR THE EAX' OPTIMIZATIONS

This section provides justifications for the simplifications of EAX' over the basic EAX mode specified by [3]. These simplifications are motivated by a desire to reduce the number of AES block encryptions required by EAX, and to reduce the amount of per-key-related storage needed by a time-optimized implementation, without significantly weakening the cryptographic strength and resistance to attack that EAX offers. The EAX' optimizations reduce the required per-key storage for a time-optimized implementation from those of unmodified EAX by $K+4B+2T$ bytes per key (88 bytes when AES-128 is the block cipher) to $K+2B$ bytes per key (48 bytes when AES-128 is the block cipher), where K is the key size of the underlying block cipher, $B$ is the block size of that block cipher, and $T$ is the desired authentication tag size, all in bytes. For space-optimized implementations, where only $K$ bytes of storage per key are required, these optimizations eliminate either three extra invocations of AES per message when <epsem-data> secrecy is used [1], or five extra invocations when it is not used.

**BACKGROUND**: Block ciphers such as AES-128 are keyed pseudo-random permutations (PRPs), which map a block-size input (e.g., 16 bytes = 128 bits) to a block-size output. The strength and ability to resist cryptanalysis of composite cryptographic modes which use this block cipher are directly related to the strength and ability to resist cryptanalysis of the underlying PRP. Analysis of a new mode must examine its impact on increasing or decreasing this strength, as well as any avenues of cryptanalytic attack that such a new mode may expose.

Consideration of the security proof of EAX, which is published in [3], indicates that the proof can be modified to account for the following optimizations. Thus the security properties of EAX' are equivalent to those of EAX. Here is the list of ways that EAX' differs from EAX and an overview of the basic rationale for each.

1. **ISSUE**: **Reduction of EAX' to two input strings –** Non-inclusion of CMACk($0^{n-1}1$ || pad(nullString) ) when the EAX input H is null.

   **JUSTIFICATION**: $\text{CMAC}_k(0^{n-1}1 \,||\, \text{pad(nullString)})$ is precisely $\text{CBC}_k(0^{127}1 \,||\, (10^{127} \text{ XOR } \text{dbl}(\text{dbl}(\text{ECB}_k(0)))))$, which is a key-dependent constant. A predictable-to-an-attacker XOR of a key-dependent constant that is the output of a keyed PRP, into a key-dependent variable that is itself a combination of keyed PRP outputs, does not strengthen the cryptanalytic resistance of the computation, because the dimensionality of the resultant composite PRP is identical to that without the inclusion. Therefore its exclusion does not weaken the cryptanalytic resistance of the computation.

2. **ISSUE**: **Exclusion of the ciphertext tag input from a null plainext string –** Conditional non-inclusion of $\text{CMAC}_k(0^{n-2}10 \,||\, \text{CTR}_k(N, \text{nullString}))$ when the EAX input $M$ is null.

   **JUSTIFICATION**: $\text{CTR}_k(N, \text{nullString})$ is the null string. Thus $\text{CMAC}_k(0^{n-2}10 \,||\, \text{pad}(\text{CTR}_k(N, \text{nullString})))$ is precisely $\text{CBC}_k(0^{n-2}10 \,||\, (10^{n-1} \text{ XOR } \text{dbl}(\text{dbl}(\text{ECB}_k(0)))))$, which is a key-dependent constant. A predictable-to-an-attacker conditional XOR of a key-dependent constant that is the output of a keyed PRP, into a key-dependent variable that is itself a combination of keyed PRP outputs, does not strengthen the cryptanalytic resistance of the computation, because the dimensionality of the resultant composite PRP is identical to that without the inclusion. Therefore the conditional exclusion of this XOR does not weaken the cryptanalytic resistance of the computation.

3. **ISSUE**: **Simplified nonce incrementation in CTR-mode processing –** Forcing two bits of the initial nonce input to a CTR-mode keyed encryption to zero, to eliminate inter-word carries, reduces the average number of messages that can be encrypted before nonce reuse without reducing the maximum message length that can be protected.

   **JUSTIFICATION**: Rogaway [6] introduced a similar optimization in his SIV combined authentication and encryption mode, which was developed after EAX and which has also been submitted to NIST. The resulting reduction in the average number of messages is at most a factor of four. However, since the other nonce bits are unpredictable to an attacker, the attacker cannot determine when nonce reuse might occur, and thus cannot exploit this potential but undetectable-to-the-attacker nonce reuse. Thus there is no practical reduction in the cryptanalytic resistance of the result.

4. **ISSUE**: **Alternate initial CMAC blocks –** Use of constants other than $0^n$, $0^{n-1}1$, and $0^{n-2}10$ as the initial

constants for the three CMAC computations of EAX.

**JUSTIFICATION**: The real requirement in EAX is that the three CMAC sequences each start with a value in the first block that is disjoint from the values of the first blocks of the other sequences, so that even if the remainder of their input sequences are identical, their outputs will not be. This will cause the initial values of the CBC sequences for each block to differ from that of the other blocks in a key-dependent way that is unpredictable to an attacker. The original EAX choices of $0^n$, $0^{n-1}1$, and $0^{n-2}10$ were arbitrary.

5. **ISSUE**: **Pre-encryption of each initial CMAC block – Inclusion** of a key-encrypted starting value in the CMAC' CBC computation, in lieu of prepending a block containing a known constant to the sequence of blocks to be processed by CBC.

   **JUSTIFICATION**: $CMAC_K$ ($J$ || $S$, $D$, $Q$) is identical to CMAC' ($K$, $E_K(J)$, $S$, $D$, $Q$). Equivalently, CMAC' ($K$, $J'$, $S$, $D$, $Q$) is identical to $CMAC_K$ ($D_K(J')$ || $S$, $D$, $Q$), where $D_K$ is the keyed decryption operator – the inverse of the keyed encryption operator $E_K$. Thus this issue reduces to issue 4.

6. **ISSUE**: **Use of *D* and *Q* for the initial CMAC blocks –** The issue is use of a derived initial value for the first block of the conditional CMAC' computation that includes a CTR-mode output, where that value is a constant multiple (in an appropriate predefined canonical GF(2) extension field) of the initial value for the first block of the other CMAC' computation, where that second initial value is the output of a keyed PRP, in contrast to requiring that the initial value for that second CMAC computation (which is conditional) be a keyed PRP output of an independent initial value.

   **JUSTIFICATION**: The CMAC' computations here always involve CBC of at least two blocks. As in issue 4, the requirement is that the initial value of that CBC sequence differs between the two sequences, in a key-dependent way that is unpredictable to an attacker. Whether the following CMAC' inputs to the two sequences are identical or not, the results of the corresponding CBC sequences are different because those first non-zero outputs of the CBC-chaining process differ for the two CBC sequences. Because the second CMAC' computation occurs only when there is Plaintext/ Ciphertext, so that the corresponding input is non-null, the results of the second CMAC' operator always differ from that of the first. This is the weakest justification of this set of six, but a careful analysis of the original security proof of EAX indicates that this optimization, which eliminates the need for time-optimized implementations to store 32 extra bytes per key, does not degrade the security of the EAX mode.

## IV. EAX' ALGORITHM

EAX' takes as input a Cleartext ($N$) and an optional Plaintext ($P$) and produce as output a Message Authentication Code ($T$) and a corresponding optional Ciphertext ($C$).

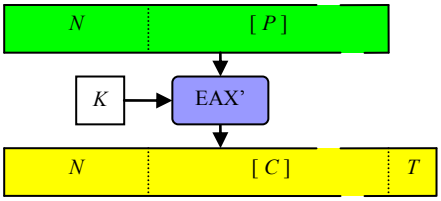


**Figure 1: EAX' Overview**

Each component of the message is defined as follows:

$N$ : Cleartext, part of the ANSI C12.22 Message that is authenticated but not encrypted for secrecy.

$P$ : Plaintext, part of the ANSI C12.22 Message which is encrypted for secrecy. When used for message secrecy, the Plaintext is composed of the <epsem-data> and the optional <padding>. <epsem-data> is the <user-information> portion of the Association Control Service Element (ACSE) [9], and thus a subset of the full application protocol data unit.

$C$ : Ciphertext, part of the ANSI C12.22 Message after encryption for secrecy, corresponding directly to the plain text P.

$T$ : Four bytes MAC (Message Authentication Code) inserted at the end of the ANSI C12.22 Message.

NOTE: To provide interoperable security it is absolutely critical that $N$, $P$, $C$, and $T$ are generated and processed in a consistent fashion. The ANSI C12.22 standard allows substantial flexibility in message transmission. In its application of ACSE, C12.22 allows for relative object identifiers. These identifiers must be expanded to absolute

identifiers for cryptographic processing. Section 5.2.4 of the ANSI C12.22 standard provides specific details regarding how a C12.22 Message is therefore canonified into a consistent standard form for cryptographic processing. *N* and *P* must be generated according to these rules. *C* and *T* are generated based on the algorithms below.

The cryptographic key, *K*, is defined as follows:

*K* : Cryptographic secret key. The key length is sized appropriate to the block cipher. The default value is set to AES-128/EAX'. The EAX' mode supports using alternate block ciphers. AES-128 has a block size of 16 bytes (128 bits).

EAX' is described in a series of algorithms which follow, where these algorithms make use of the following operators and notations:

**Table 1: Operators and notations used**

| ***Operation*** | **Definition** |
|---|---|
| $X \leftarrow Y$ | Y assigned to X |
| $X$ & $Y$ | Bitwise AND of X and Y where X and Y and the result are the same size |
| $X$ XOR $Y$ | Bitwise exclusive-or of X and Y where X and Y and the result are the same size |
| $X$ $\text{XOR}_{\text{end}}$ $Y$ | Bitwise exclusive-or (XOR) of Y after alignment with the end of string X where X is longer than Y and the result is the same size as X. This operation is exactly equivalent to X XOR ($0^{\text{size(X)-size(Y)}}$ \|\| Y). |
| size($X$) | Length in bits of X |
| $AES_K(X)$ | Encryption of X using AES with key K. X, K and the result are all exactly n bits long. |
| $X \,\|\|\, Y$ | The concatenation of strings X and Y |
| $X$ [first $m$ bits] | Most significant m bits of X |
| msb($X$) | Most significant single bit of X |
| $X << 1$ | Left shift of X by one position (i.e., with a carry-in of 0) |
| ceiling($X$) | Ceiling function, which returns the smallest integer not less than $X$ |
| n | The block size in bits of the underlying block cipher (for AES-128, n = 128) |
| $0^m$ | m bits of 0. (i.e., $0^4 10^3$ would be 00001000) |
| $1^m$ | m bits of 1. (i.e., $1^4 01^2 0$ would be 11110110) |

*A. Algorithm dbl(X)*

For this algorithm the minimal weight irreducible polynomial $r_b$ is defined exactly as in [4].

```
-- modulo the first minimal weight irreducible polynomial of
     degree n
10 define r_b as 0^120 10000111   -- for n = 128
11 if msb(X) = 0       -- most significant bit
12 then return X << 1
13 else return (X << 1) XOR r_b
```

*B. Algorithm deriveKeyDependentConstants(K)*

```
20 L ← AES_K(0^n)
     -- For the AES-128 block cipher n = 128
21 D ← dbl(L)
     -- D is double the value of L in an appropriate field, as
     defined by r_b
22 Q ← dbl(D)
     -- Q is quadruple the value of L
     -- dbl(dbl(L))
     -- in the same field
23 return D and Q
```

*C. Algorithm pad (S, D, Q)*

```
30 if (size(S) > 0) and (size(S) mod n = 0)   -- n equals Cipher
     block size (in bits)
31 then return S XOR_end D   -- XORs into end of string
32 else return (S || 10^(n−1−(size(S) mod n))) XOR_end Q   -- XORs
     into end of string; applies to null string
```

*D. Algorithm CBC'$_K$ (t, S)*

```
40 Let S_1 ... S_m ← S where size(S_i) = n   -- n equals Cipher
     block size (in bits)
41 C_0 ← t
42 for i ← 1 to m
43 do C_i ← AES_K(S_i XOR C_i−1)
44 return C_m
```

The following definition of CMAC' is derived from [4]. It has been recast into nomenclature similar to that used in the EAX papers and the syntax used by the ANSI C12.22 standard.

*E. Algorithm CMAC'K (t, S, D, Q)*

```
50 return CBC'_K (t, pad (S, D, Q))
```

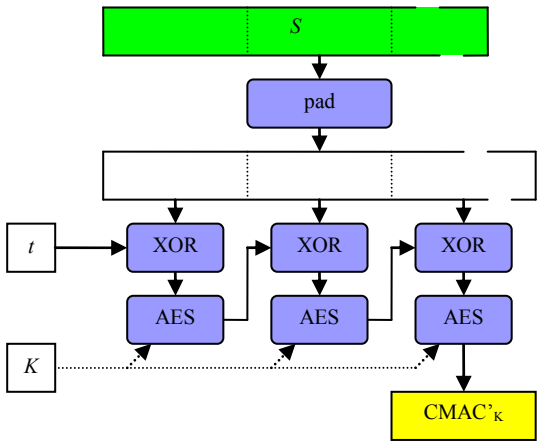


**Figure 2: Computation algorithm of CMAC'$_K$**

The following definition of CTR chaining mode is derived from [5]. It has been recast into nomenclature similar to that used in the EAX papers and the syntax used in the ANSI C12.22 standard.

### F. *Algorithm CTR'$_K$ (N, S)*

Note: For this algorithm, the counter *Ctr* can be thought of as a single *n*-bit number represented as *m* bytes with the most significant byte placed first, (byte 0) and least significant byte placed last (byte *m*-1). By contrast, all the other functions that implement EAX', including AES, are defined with the least significant byte placed first. This was done as an acknowledgement of extant existing practice in implementations of CTR mode as defined in [5].

60 $m \leftarrow ceiling(\text{size}(S)/n)$
61 $Ctr \leftarrow N \,\&\, 1^{n-32}\,01^{15}\,01^{15}$
-- Optimization, to avoid inter-word carries.
62 -- For AES, n=128 and $1^{n-32}\,01^{15}\,01^{15}$ resolves to
63 -- FFFF FFFF FFFF FFFF FFFF FFFF 7FFF 7FFF$_H$
64 $Pad \leftarrow AES_K(Ctr) \,||\, AES_K(Ctr+1) \,||\, \ldots \,||\, AES_K(Ctr+m-1)$
65 **return** $S$ XOR $Pad$ [first size($P$) bits]

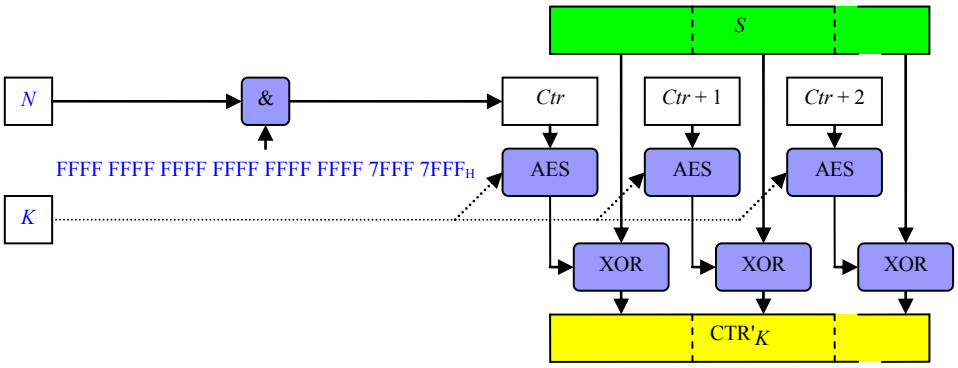


**Figure 3: Computation algorithm of CTR'$_K$**

### G. *Algorithm EAX'.Encrypt$_K$ (N, P)*

70 deriveKeyDependentConstants($K$)
-- need be done only once per new key value
71 $Tag \leftarrow \underline{N} \leftarrow \text{CMAC}_K(D, N, D, Q)$
-- EAX' optimization, first $D$ used instead of $E_K(0^n)$
72 **if** size($P$) > 0
-- EAX' optimization, no contribution of a key-dependent constant for an empty $C$
73 **then**
$C \leftarrow$ **nullString**
74 **else**
75 $C \leftarrow \text{CTR'}_K(\underline{N}, P)$
76 $Tag \leftarrow Tag$ XOR $\text{CMAC}_K(Q, C, D, Q)$ }
-- EAX' optimization, first $Q$ used instead of $E_K(0^{n-2}10)$
77 $T \leftarrow Tag$ [first *32* bits]
78 **return** $C$ and $T$

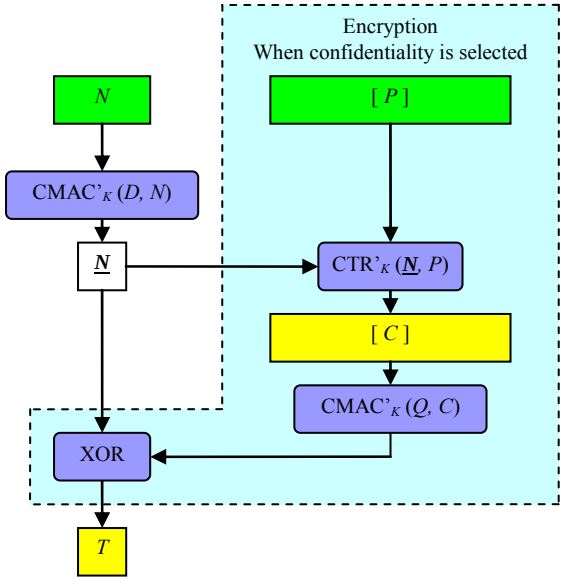


**Figure 4: Encryption algorithm when confidentiality is selected**

*H. Algorithm EAX'.Decrypt$_K$ (N, C, T)*

```
80 deriveKeyDependentConstants(K)
     -- need be done only once per new key value
81 Tag ← N ← CMAC_K (D, N, D, Q)
     -- EAX' optimization, first D used instead of E_K(0^n)
82 if size(C) = 0 then
     -- EAX' optimization, no contribution of a key-dependent
     constant for an empty C
83     P ← nullString
84     if T ≠ Tag [first 32 bits] then
         return INVALID
       else return P -- message is VALID
85 else
86     Tag ← Tag XOR CMAC_K (Q, C, D, Q)
     -- EAX' optimization, first Q used instead of E_K(0^(n-2)10)
87     if T ≠ Tag [first 32 bits] then return INVALID
88     P ← CTR'_K (N, C)
     -- decrypt Ciphertext only when tags match
89 return P -- message is VALID
```

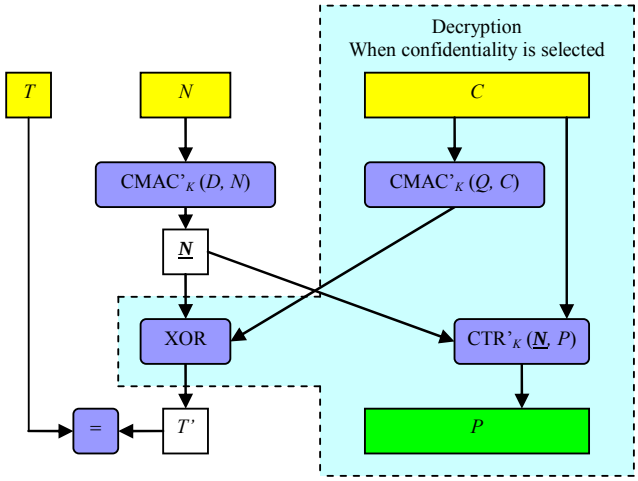


**Figure 5: Decryption algorithm when confidentiality is selected**

## V. Test Vectors

The following test vectors are for use when the underlying block cipher is AES-128. All inputs and outputs are shown as the hexadecimal representation of sequential bytes.

### A. Test Vector 1

#### 1) Inputs:

**Key:**
```
01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08
```
**Plaintext:**
```
a2 0c 06 0a 60 7c 86 f7 54 01 16 00 17 02 a7 03
02 01 04 a8 03 02 01 02 ac 0f a2 0d a0 0b a1 09
80 01 02 81 04 48 e9 93 88 be 19 28 17 81 15 9a
a6 0d 06 0b 60 7c 86 f7 54 01 16 00 17 82 11 02
48 e9 93 88
```
**Cleartext:**
```
54 45 4d 50 0b 40 00 07 00 05 1a 00 00 02 00 e4
```

#### 2) Outputs:

**Ciphertext:**
```
40 31 cc 95 7d 4e df 9a 35 7f 3d b0 fa 9f e8 38
```
**MAC:**
```
65 55 c0 29
```

### B. Test Vector 2

#### 1) Inputs:

**Key:**
```
01 02 03 04 05 06 07 08 01 02 03 04 05 06 07 08
```
**Plaintext:**
```
a2 0c 06 0a 60 7c 86 f7 54 01 16 00 7b 02 a7 03
02 01 04 a8 03 02 01 02 ac 0f a2 0d a0 0b a1 09
80 01 02 81 04 48 f3 d2 f8 be 19 28 17 81 15 9a
a6 0d 06 0b 60 7c 86 f7 54 01 16 00 7b 82 11 02
48 f3 d2 f8
```
**Cleartext:**
```
54 45 4d 50 0b 40 00 07 00 05 1a 00 00 02 00 e4
```

#### 2) Outputs:

**Ciphertext:**
```
8d 2f bb 7a 0a 8c 4d 40 ed aa 10 a4 64 31 c9 b8
```
**MAC:**
```
fe c6 d9 e8
```

### C. Test Vector 3

#### 1) Inputs:

**Key:**
```
10 20 30 40 50 60 70 80 90 a0 b0 c0 d0 e0 f0 00
```
**Plaintext:**
```
a2 0e 06 0c 60 86 48 01 86 fc 2f 81 1c aa 4e 01
a8 06 02 04 39 a0 0e bb ac 0f a2 0d a0 0b a1 09
80 01 00 81 04 4b ce e2 c3 be 25 28 23 81 21 88
a6 0a 06 08 2b 06 01 04 01 82 85 63 00 4b ce e2
c3
```
**Cleartext:**
```
17 51 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 00 00 03 30 00 01
```

#### 2) Outputs:

**Ciphertext:**
```
9c f3 2c 7e c2 4c 25 0b e7 b0 74 9f ee e7 1a 22
0d 0e ee 97 6e c2 3d bf 0c aa 08 ea
```
**MAC:**
```
00 54 3e 66
```

### D. Test Vector 4

#### 1) Inputs:

**Key:**
```
66 24 c7 e2 30 34 e4 03 6f e5 cb 3a 8b 5d ab 44
```
**Plaintext:**
```
a2 11 06 0f 2b 06 01 04 01 82 85 63 8e 7f 85 f1
c2 4e 01 a8 06 02 04 2b c8 1a a1 ac 0f a2 0d a0
0b a1 09 80 01 00 81 04 4b 97 d2 cc be 39 28 37
81 35 88 a6 09 06 07 2b 06 01 04 82 85 63 00 4b
97 d2 cc
```
**Cleartext:**
```
17 51 30 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 30 30 30 30 00 00 03 30 00 01 03 30 00 78
03 30 00 79 03 30 00 7a 03 30 00 7b 03 30 00 7d
```

#### 2) Outputs:

**Ciphertext:**
```
be b0 98 9f ad b0 20 eb 72 ba 46 35 3c c0 a2 ac
2a 00 7a 10 1a fe ba f9 68 0d 3b 96 59 f9 91 12
1b 86 5f 25 4f 6a c9 2c dd 21 3d 31 e3 c4 d2 ca
```
**MAC:**
```
e6 f8 9b 6d
```

## VI. IPR Statement

NEMA gives permission to NIST to reproduce a description of the EAX' algorithm which is Appendix I of the ANSI C12.22-2008 standard.

The origin of the algorithm is as described herein and NEMA makes no IPR claims to the EAX' algorithm.

## VII. Acknowledgement

This EAX' description is derived from [3]. Its presentation is based on the algorithms specified in Figures 1 - 3 of that paper, recast into a syntax appropriate to this Standard. The authors of the referenced EAX papers [2][3], generously granted permission to use those figures. While the material below is not a direct reproduction of those figures, they are clearly the motivation for this presentation.

NEMA has graciously given permission to the authors to reproduce content from the ANSI C12.22 standard in this report to support the full consideration by NIST of EAX'.

## References

[1] "American National Standard Protocol Specification For Interfacing to Data Communication Networks", ANSI C12.22-2008 / IEEE P1703 / MC1222: Annex I, "EAX' description"

[2] M. Bellare, P. Rogaway and D. Wagner, Authenticated Encryption with Associated Data (AEAD) algorithm designed to simultaneously protect both authentication and privacy of messages, as described in "A Conventional Authenticated-Encryption Mode", April 13, 2003, available from http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf.

[3] M. Bellare, P. Rogaway, and D. Wagner, "The EAX Mode of Operation, A Two-Pass Authenticated-Encryption Scheme Optimized for Simplicity and Efficiency", January 18 2004, available from http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf.

[4] "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", NIST SP 800-38B, available from http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[5] "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST SP 800-38A, available from http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[6] P. Rogaway, T. Shrimpton, "The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption", August 20, 2007, available from http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/siv/siv.pdf

[7] "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication", NIST SP 800-38C, available from http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf.

[8] "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38C, available from http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[9] ISO/IEC 10035-1 Information Technology—Open Systems Interconnection—Connectionless Protocol for the Association Control Service Element: Protocol Specification, 1995

[10] ISO/IEC 8825-1 Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), July 2008